Merge step in Merge sort

Suresh Podeti

2 min read · May 15, 2023

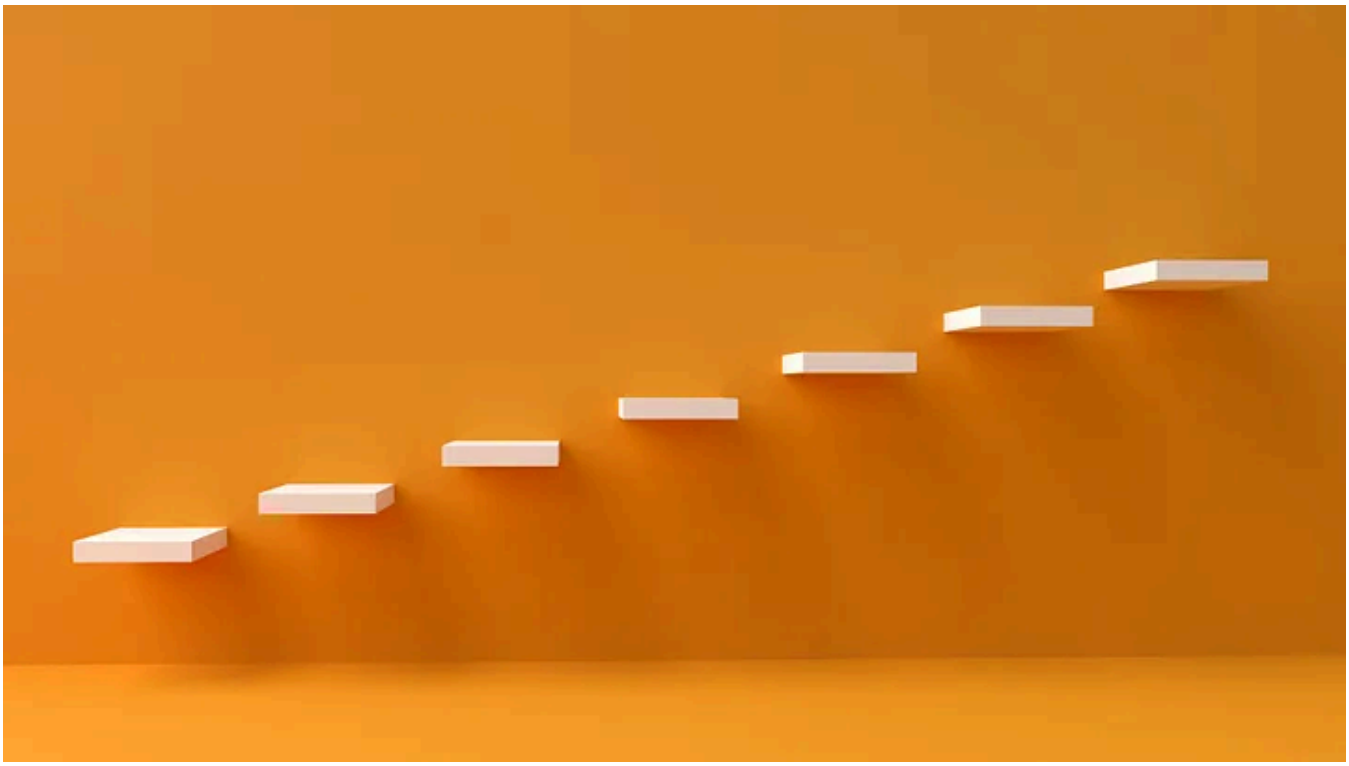
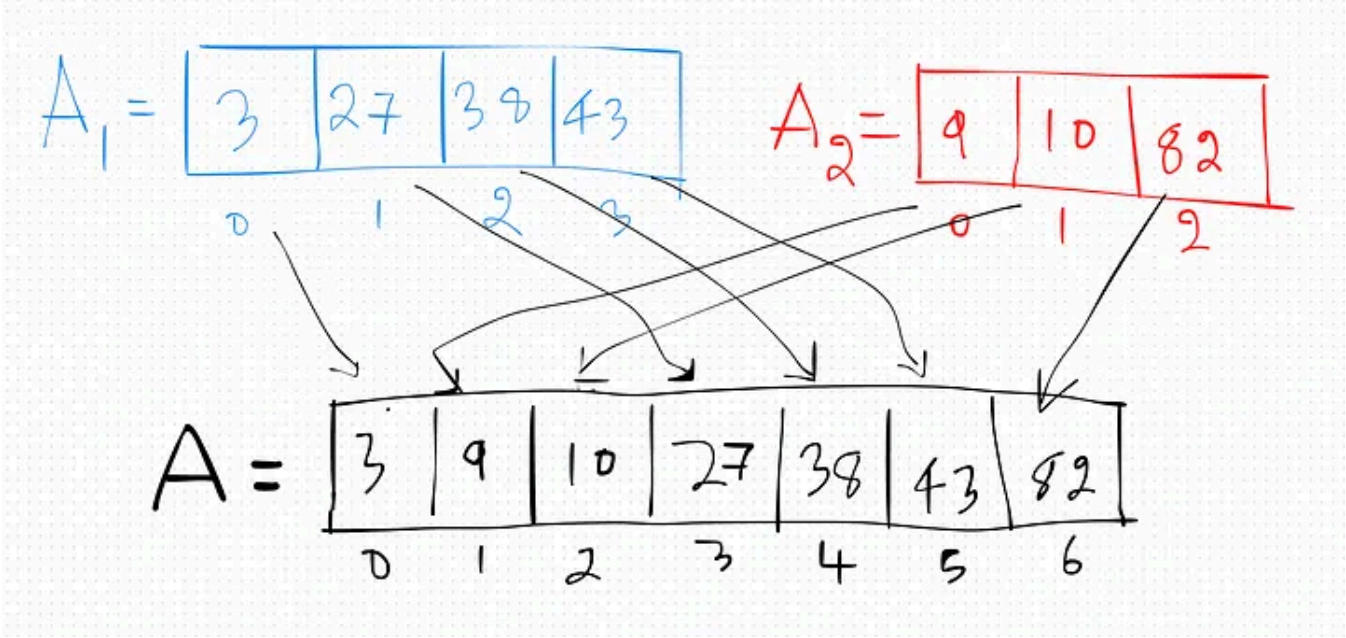


Photo by Kanhaiya Sharma on Unsplash

Merge sort is a **divide-and-conquer** algorithm based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.

In this article, we will focus more on **merge step** of merge sort algorithm, which is **merging** the two sorted halves into a single sorted array. Given two sorted arrays A1 = {3, 27, 38, 43}, and A2= {9, 10, 82} merge step will merge two sorted arrays into a single sorted array A = {3,9,10,27,38, 43, 82}. Time complexity of the merge step is ~ **O(n)** where n is the size of the merged array.



Algorithm

Let i, j, and k are pointing to array A1, A2, and A respectively.

1. Initialise i = j=k=0
2. Check if A1[i] ≤ A2[j], A[k++] = A1[i++]
3. else, A[k++] = A2[j++]
4. Continue, loop if i<A1.length && j< A2.length
5. Check if i<A1.length, loop until i≥A1.length and A[k++] = A1[i++]
6. Check if j<A2.length, loop until j≥A2.length and A[k++] = A2[j++]

Following is merge logic written in C++:

```
#include <iostream>

using namespace std;

void merge(int *a1, int *a2, int m, int n){
    int a[n+m];

    int i,j,k;

    i=j=k=0;

    while(i<m&&j<n){
        if(a1[i]<a2[j]){
```

```
            a[k++] = a1[i];
            i++;
        }else{
            a[k++] = a2[j];
            j++;
        }
    }

    while(i<m) a[k++] = a1[i++];

    while(j<n) a[k++] = a2[j++];


    for(int i=0;i<m+n;i++) cout << a[i] <<" ";
    cout << endl;
}

int main()
{
    int a1[] = {3, 27, 38, 43};
    int a2[] = {9, 10, 82};

    int m = sizeof(a1)/sizeof(a1[0]);
    int n = sizeof(a2)/sizeof(a2[0]);

    merge(a1, a2, m, n);
    return 0;
}
```

This is all for this article.

Merge Sort | Algorithms | Datastrucutre | Ascending Order | Sorting Algorithms

Edit profile

Written by Suresh Podeti

1.2K Followers